11 | 矢量运算：Java的机器学习要来了吗？

2021-12-08 范学雷

《深入剖析Java新特性》 课程介绍 >

讲述：范学雷
时长 07:07 大小 6.53M

你好，我是范学雷。今天，我们讨论 Java 的矢量运算。

Java 的矢量运算，我写这篇文章的时候还在孵化期，还没有发布预览版。我们之所以选取了这样一个还处于孵化期的技术，主要是因为这个技术代表了 Java 语言发展的一个重要方向，在未来一定会有着重要的影响。早一点了解这样的技术，除了扩展视野之外，还能够帮助我们制定未来几年要学习或者要使用的技术路线。

我们从阅读案例开始，看一看没有矢量运算的时候，Java 是怎么支持科学计算的；然后我们再看看矢量运算能够带来什么样的变化。

阅读案例

我想，你对线性方程（或者说一次方程）一定不陌生。一般情况下，我们可以把线性方程表述成下面的形式。

$$y = a_0x_0 + a_1x_1 + a_2x_2 + \ldots + a_{n-1}x_{n-1}$$

其中 a_0，a_1，a_{n-1} 表示的是常数，x_0，x_1，x_{n-1} 表示的是变量，而 y 就表示 a_i 和 x_i 的组合结果。n 表示未知变量的数目，通常，我们也把它称为方程的维度。

如果给定方程式右边的常数和变量，我们就能计算出方程式左边的 y 数值了。那么，该怎么用代码表示这个方程式呢？我们可以把 a_0，a_1，a_{n-1} 表示的常数放到一个数组里，把 x_0，x_1，x_{n-1} 表示的变量放到另外一个数组里。下面的代码里，变量 a 和 x 就可以用来表示一个有四个维度的一次方程组。

复制代码

```
static final float[] a = new float[] {0.6F, 0.7F, 0.8F, 0.9F};
static final float[] x = new float[] {1.0F, 2.0F, 3.0F, 4.0F};
```

能用 Java 的变量来表示一次方程，我们也就能够计算线性方程的结果了。下面的代码，就是一个实现的办法。

复制代码

```
private static Returned<Float> sumInScalar(float[] a, float[] x) {
    if (a == null || x == null || a.length != x.length) {
        return new Returned.ErrorCode(-1);
    }
    float[] y = new float[a.length];
    for (int i = 0; i < a.length; i++) {
        y[i] = a[i] * x[i];
    }
    float r = 0F;
    for (int i = 0; i < y.length; i++) {
        r += y[i];
    }
    return new Returned.ReturnValue<>(r);
}
```

在上面的代码里，我们先计算 a_i 和 x_i 的乘积，然后再计算乘积结果的总和。其中的乘法运算，就是我们常说的标量运算。为了方便讨论，我把乘法运算的代码单独拿出来，粘贴在下面。

复制代码

```
float[] y = new float[a.length];
for (int i = 0; i < a.length; i++) {
    y[i] = a[i] * x[i];
}
```

如果我们仔细观察线性方程就会发现，对于每一个纬度，a_i 和 x_i 是互不影响的， 当然它们的乘积也是互不影响的。既然每个维度的计算都互不影响，那么我们能不能并行计算呢？

矢量运算

Java 的矢量运算就是使用单个指令并行处理多个数据的一个尝试（单指令多数据，Single Instruction Multiple Data）。

在现代的微处理器（CPU）中，一个控制器可以控制多个平行的处理单元；在现代的图形处理器（GPU）中呢，更是拥有强大的并发处理能力和可编程流水线。这些处理器层面的技术，为软件层面的单指令多数据处理提供了物理支持。Java 矢量运算的设计和实现，也是希望能够借助现代处理器的这种能力，提高运算的性能。

为了使用单指令多数据的指令，我们需要把不同数据的运算独立出来，让并行运算成为可能。而数学里的矢量运算，恰好就能满足这样的要求。

如果使用矢量，我们可以把线性方程表述成下面的形式（使用向量的数量积形式）：

$$y' = ax$$

$$y = \sum_{0}^{n-1} y'_i$$

其中，a，x 和 y' 是三个 n 维的矢量。

$$a = [a_0, a_1, a_2, \ldots a_{n-1}]$$

$$y' = [y'_0, y'_1, y'_2, \ldots, y'_{n-1}]$$

好了，现在我们可以看看 Java 是怎么表达矢量的了。下面代码里的变量 a，和前面阅读案例里 a 是一样的，它以数组的形式表示；变量 va，就是变量 a 的矢量表达形式。fromArray 这个方法，可以把一个数组变量，转换成一个矢量的变量。

复制代码

```
static final float[] a = new float[] {0.6F, 0.7F, 0.8F, 0.9F};
static final FloatVector va =
        FloatVector.fromArray(FloatVector.SPECIES_128, a, 0);

static final float[] x = new float[] {1.0F, 2.0F, 3.0F, 4.0F};
static final FloatVector vx =
        FloatVector.fromArray(FloatVector.SPECIES_128, x, 0);
```

有了表示矢量的办法，我们就可以试着使用矢量运算的办法，来计算线性方程的结果了。下面的代码，就是一个简化了的实现。

复制代码

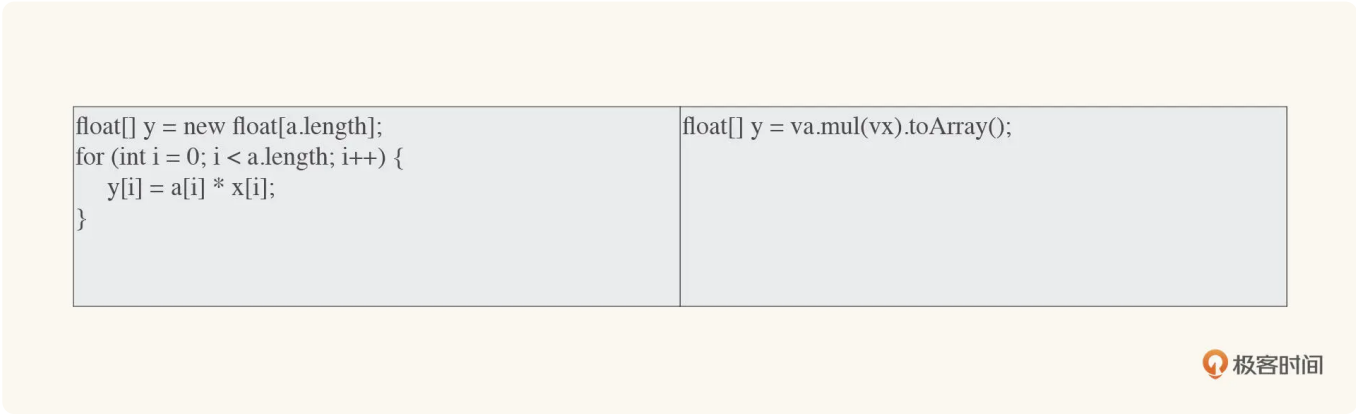
```
private static Returned<Float> sumInVector(FloatVector va, FloatVector vx) {
    if (va == null || vx == null || va.length() != vx.length()) {
        return new Returned.ErrorCode(-1);
    }

    // FloatVector vy = va.mul(vx);
    float[] y = va.mul(vx).toArray();

    float r = 0F;
    for (int i = 0; i < y.length; i++) {
        r += y[i];
    }
    return new Returned.ReturnValue<>(r);
}
```

这个运算的关键部分是其中的矢量运算，也就是下面这行代码。

```
1 FloatVector vy = va.mul(vx);
```

复制代码

和上面的标量运算的办法相比，矢量运算的代码精简了很多。这是矢量运算的第一个优点。但它的优点还不止于此。



飙升的性能

我们前面提到，Java 矢量运算的设计，主要是为了性能。 那么，性能的提升能有多大呢？我自己做了一个性能测试。虽然这个特性还处于孵化期，但是它的性能测试结果还是很令人振奋的。 就上面这个简单的、四维的矢量来说，和我们在阅读案例里使用的标量运算相比，矢量运算的性能提高了足足有 10 倍。

复制代码

```
1 Benchmark                          Mode  Cnt          Score           Error  Uni
2 VectorBench.scalarComputation     thrpt   15  180635563.597 ±  30893274.582  ops
3 VectorBench.vectorComputation     thrpt   15 1839556188.443 ± 153876900.442  ops
```

对于一个还处于孵化阶段的实现来说，这么大的性能提升是有点超出预料的。

在密码学和机器学习领域，通常需要处理几百甚至几千维的数据。一般情况下，为了能够使用处理器的计算优势，我们经常需要特殊的设计以及内嵌于 JVM 的本地代码来获得硬件加速。这样的限制，让普通代码的计算很难获得硬件加速的好处。

希望成熟后的 Java 矢量运算，能在这些领域有出色的表现，让普通的代码获得处理器的单指令多数据的强大运算能力。毕竟，只有单指令多数据的优势能够被普通的 Java 应用程序广泛使用，Java 才能在机器学习、科学计算这些领域获得计算优势。

如果从机器学习在未来的重要性来说，Java 在科学计算领域的拓展来得也许正是时候。

总结

好，到这里，我来做个小结。前面，我们讨论了 Java 的矢量运算这个尚处于孵化阶段的新特性，对 Java 的矢量运算这个新特性有了一个初始的印象。

如果 Java 矢量运算成熟起来，许多领域都可以从这个新特性中受益，包括但是不限于机器学习、线性代数、密码学、金融和 JDK 本身的代码。

这一次学习的主要目的，就是让你对矢量运算有一个基本的印象。这样的话，如果你的代码里有大量的数值计算，也许可以考虑在将来使用矢量运算获得硬件的并行计算能力，大幅度提高代码的性能。

由于矢量运算尚处于孵化阶段，目前我们还不需要学习它的 API，知道 Java 有这个发展方向，并且能够思考你的代码潜在的改进空间就足够了。知道了这个方向，等 Java 矢量运算正式发布的时候，你就可以尽早地改进你的代码，从而获得领先的优势了。

如果面试中聊到了数值计算的性能，你应该知道有矢量运算这么一个潜在的方向，以及“单指令多数据”这么一个术语。

思考题

其实，今天的这个新特性，是练习使用 JShell 快速学习新技术的一个好机会。使用阅读案例里提供的数据，你能够使用 JShell，快速地表示出下面的这个矢量吗？

$$y' = ax$$

需要注意的是，要想使用孵化期的 JDK 技术，需要在 JShell 里导入孵化期的 JDK 模块，就像下面的例子这样。

复制代码

```
$ jshell --add-modules jdk.incubator.vector -v
|  Welcome to JShell -- Version 17
|  For an introduction type: /help intro

```

```
5 jshell> import jdk.incubator.vector.*;
```

欢迎你在留言区留言、讨论，分享你的阅读体验以及你的设计和代码。我们下节课见！

注：本文使用的完整的代码可以从GitHub下载，你可以通过修改GitHub上review template代码，完成这次的思考题。如果你想要分享你的修改或者想听听评审的意见，请提交一个 GitHub 的拉取请求（Pull Request），并把拉取请求的地址贴到留言里。这一小节的拉取请求代码，请在矢量运算专用的代码评审目录下，建一个以你的名字命名的子目录，代码放到你专有的子目录里。比如，我的代码，就放在 vector/review/xuelei 的目录下面。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 1 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 10 | Flow，是异步编程的终极选择吗？

下一篇 12 | 外部内存接口：零拷贝的障碍还有多少？

精选留言 (3)

写留言

aoe
2021-12-08

看个热闹

展开

1

哦吼掉了
2021-12-09

遇到的问题：程序包 jdk.incubator.vector 已在模块 jdk.incubator.vector 中声明, 但该模块

不在模块图中；
咨询的问题：如果稍微大一点的java脚本(一堆测试代码)，能使用jshell导入执行么？

展开

作者回复: 第一个问题，我也不知道该怎么办。我使用的是IDEA，花费了很长时间，我也没搞清楚IDEA是怎么支持孵化期的特性的。有经验的小伙伴们帮帮忙。

第二个问题，我没有看到为什么不可以，应该和脚本大小没有关系。

共 2 条评论 >

过去 Sword 将...

2021-12-08

老师，能否和之前那样说一下新特性在哪个版本以预览版出来的呢？个人电脑装了17，但是刚刚我在公司电脑jdk11输入FloatVector发现并没有

展开

作者回复: 嗯，要是说一下就好了。 孵化期的特性， 建议使用最新版本的。

共 2 条评论 >